MPTK V2.0

Migration Helper

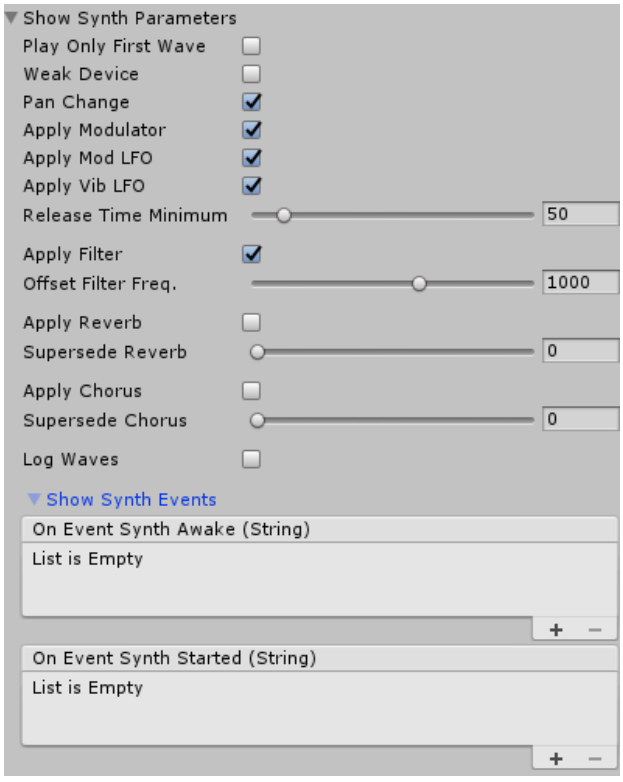
Follow precisely the migration process!

The new version is a complete rewrite of MPTK !!!

- A new synthesizer engine based on the wonderful fluidsynth: http://www.fluidsynth.org/
  - Converted to C#
  - Adapted to the Unity AudioSource capabilities
  - Adapted to be able running on weak device
  - This new version is closer to the SoundFont standard:
    - Generator
    - Modulator
    - Envelope
    - LFO
    - Effect
    - ….
- A new format for the internal SoundFont.
  - Size divided by 20
  - Time to load divided with the same ratio!
    - Example: the SF GeneralUser_GS_SoftSynth_v144 was loaded in 1 second before, now the SF is loaded in 0.04 second.

So, I encourage you to migrate. It could be a little works for you because some changes has been done but you will get a more reliable and interesting SoundFont synthetizer.

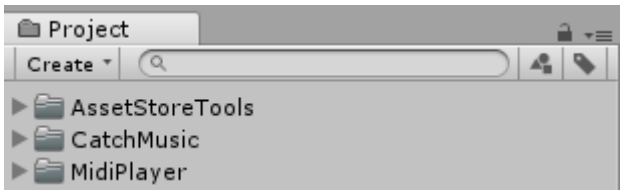
See below the new synth parameters panel:



Migration to V2

First of all: **make a strong backup of your works and check it!**

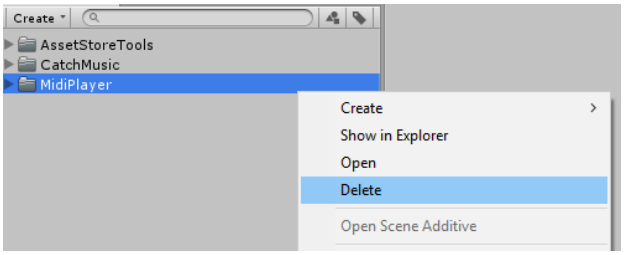
Classically, MidiPlayer is set apart of your project. This is recommended by Unity to enjoy version update. Something like that:



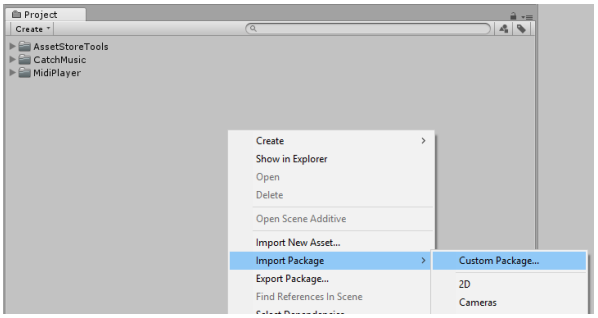
Delete the entire current version of MidiPlayer because some older class will remains in the MidiPlayer folders (Unity import don't removed these files) and you will get tricky issues.

But before, backup your midi file from Resources/MidiDB.

- There is no need to backup SoundFont as the new version is not compatible with the older. You will have to reimport your SoundFont files.
- If you have done changes in MPTK scripts (this is not recommended!), have a list somewhere of these changes to reproduces them in the new version.

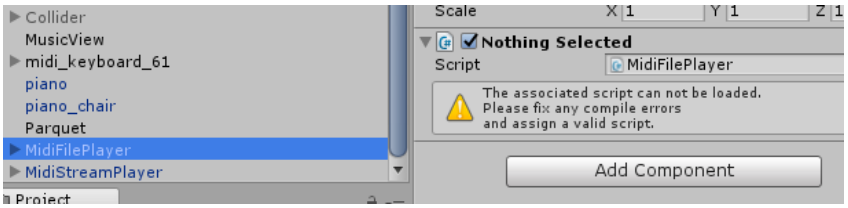


Import the new MPTK package:



Carefully, select what you need to import to avoid updating your own project. For example, unselect "Scene For Demo". There is no need in your project.

For each scenes of your project, old Prefab are to be updated:



- Remove the MPTK Prefab you are using in the hierarchy: MidiFilePlayer, MidiStreamPlayer, MidiExternalPlay, MidiListPlayer.
- Replace by the new Prefab (same name in Project view)
- If need, don’t forget to update in your project the references to the prefabs in your others gameobjects.

Some class and methods are changed in the new version. If you use them, you will have to correct all the compiler errors:

Class MidiNote and MPTKNote replaced and merged with the more Midi compliant class MPTKEvent.

Now, MPTKEvent can contains all kind of Midi events, not only note on.

```
/// <summary>
    /// Midi Event class for MPTK. Usage to generate Midi Music with MidiStreamPlayer or
    /// to read midi events from a Midi file with MidiLoad
    /// or to recevice midi events from MidiFilePlayer OnEventNotesMidi.
    /// </summary>
    public class MPTKEvent
    {
        /// <summary>
        /// Time in Midi Tick (part of a Beat) of the Event since the start of playing
        /// the midi file. This time is independant of the Tempo or Speed.
        /// Not used for MidiStreamPlayer.
        /// </summary>
        public long Tick;

        /// <summary>
        /// Midi Command code. Defined the type of message (Note On, Control Change, Patch Change...)
        /// </summary>
        public MPTKCommand Command;

        /// <summary>
        /// Controller code. When the Command is ControlChange, contains the code for the controller
        /// to change (Modulation, Pan, Bank Select ...).
        /// Value will contains the value of the controller.
        /// </summary>
        public MPTKController Controller;

        /// <summary>
        /// MetaEvent Code. When the Command is MetaEvent,
        /// contains the code of the meta event (Lyric, TimeSignature, ...). .
        /// Info will contains the value of the meta.
        /// </summary>
        public MPTKMeta Meta;

        /// <summary>
        /// Information hold by textual meta event when Command=MetaEvent
        /// </summary>
        public string Info;

        /// <summary>
        /// Contains a value between 0 and 127 in relation with the Command. For:
        ///! @li @c   Command = NoteOn then Value contains midi note
        ///! @li @c   Command = ControlChange then Value contains controller value
        ///! @li @c   Command = PatchChange then Value contains patch value
        /// </summary>
        public int Value;

        /// <summary>
        /// Midi channel fom 0 to 15 (9 for drum)
        /// </summary>
        public int Channel;

        /// <summary>
        /// Velocity between 0 and 127
        /// </summary>
        public int Velocity;

        /// <summary>
        /// Duration of the note in millisecond
        /// </summary>
        public double Duration;

        /// <summary>
```

```
/// Duration of the note in Midi Tick.
/// MidiFilePlayer.MPTK_NoteLength can be used to convert this duration.
/// Not used for MidiStreamPlayer.
/// https://en.wikipedia.org/wiki/Note_value
/// </summary>
public int Length;

/// <summary>
/// Note length as https://en.wikipedia.org/wiki/Note_value
/// </summary>
public enum EnumLength { Whole, Half, Quarter, Eighth, Sixteenth }

/// <summary>
/// List of voices associated to this Event for playing a NoteOn event.
/// </summary>
public List<fluid_voice> Voices;
```

Example of using the MPTKEvent class to detect a note: check the type of event (here a note on), get the value of note with Value (.Midi was used in the previous version)

```
foreach (MPTKEvent note in notes)
       if (note.Command== MPTKCommand.NoteOn)
              if (note.Value >= 40 && note.Value <= 127)
       …
```

These methods are obsoletes in the MPTKEvent class:

`public void Play(MidiStreamPlayer streamPlayer)`

➔ Replaced by `MPTK_PlayEvent(MPTKEvent pnote)` from class `MidiStreamPlayer`

`public void Stop()`

➔ Replaced by `MPTK_StopEvent(MPTKEvent pnote)` from class `MidiStreamPlayer`

Have a look to the demo TestMidiStream (TestMidiStream.cs source file):

```
private void PlayOneNote()
{
    StopOneNote();

    NotePlaying = new MPTKEvent()
    {
        Command = MPTKCommand.NoteOn,
        Value = CurrentNote,
        Channel = StreamChannel,
        Duration = 9999999, // 9999 seconds but stop by the new note. See before.
        Velocity = Velocity // Sound can vary depending on the velocity
    };
    midiStreamPlayer.MPTK_PlayEvent(NotePlaying);
}

private void StopOneNote()
{
    if (NotePlaying != null)
    {
        // Stop the note (method to simulate a real human on a keyboard
        // because duration is unknown when note is triggered.
        midiStreamPlayer.MPTK_StopEvent(NotePlaying);
        NotePlaying = null;
    }
}
```

Class ImSoundFont

- Description is deprecated and not replaced

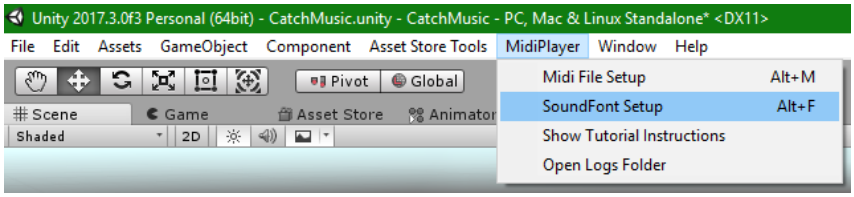
Class MidiFilePlayer

- New parameters for the EndPlay event.

```
/// <summary>
/// Event fired by MidiFilePlayer when a midi is ended when reach end or stop by
MPTK_Stop or Replay with MPTK_Replay
/// The parameter reason give the origin of the end and the name of the midi
/// </summary>
public void EndPlay(string midiname, EventEndMidiEnum reason)
{
        Debug.LogFormat("End Play Midi {0} reason:{1}", midiname, reason);
}
```

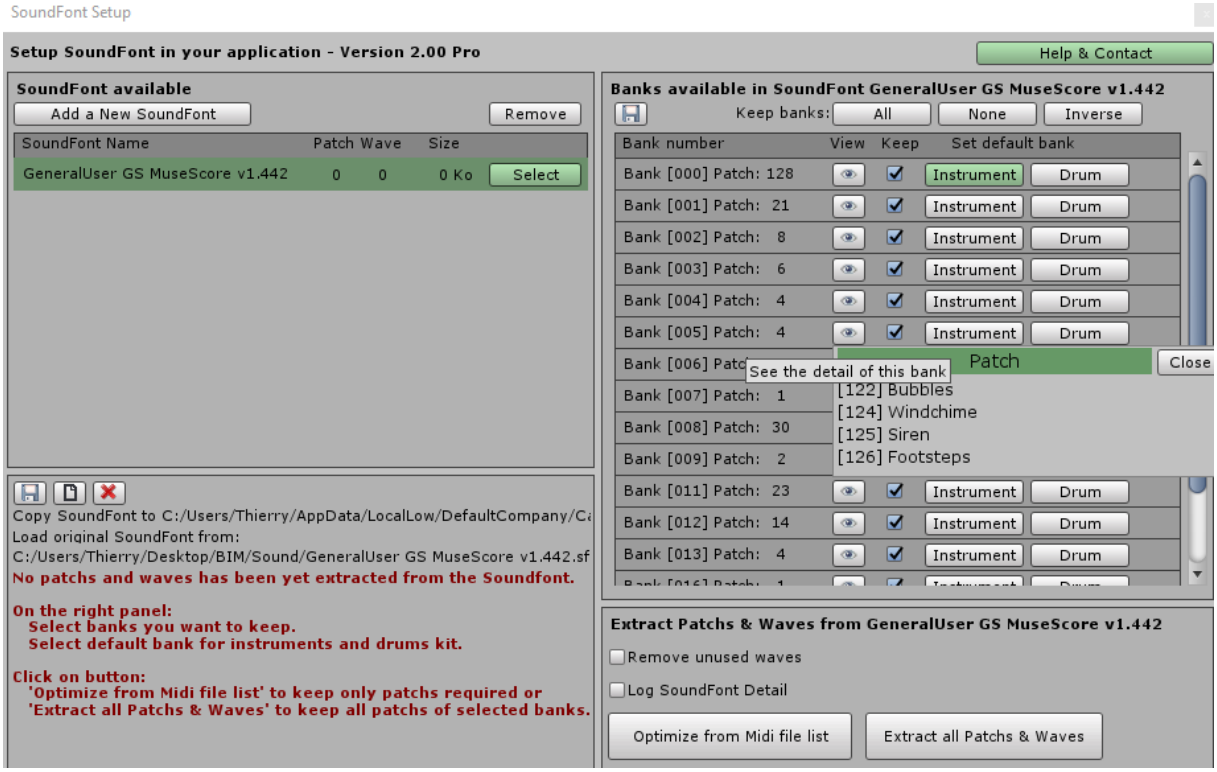
When all compiler errors are corrected, you are ready to import the new soundfont:

Now, MPTK get a dedicated menu:



The new soundfont setup window:

- Select the banks you are interested for keep
- Set the default bank for instrument and drum



Extract all patchs or optimize depending on your Midi list (Pro version) and you will be ready!